6th World Congresses of Structural and Multidisciplinary Optimization
Rio de Janeiro, 30 May - 03 June 2005, Brazil

A new algorithm to identify all global maximizers based on simulated annealing

Ana I.P.N. Pereira[1], Edite M.G.P. Fernandes[2*]

[1]apereira@ipb.pt, Polytechnic Institute of Braganca, Braganca, Portugal
[2]emgpf@dps.uminho.pt, University of Minho, Braga, Portugal

1. Abstract
In this work we consider the problem of finding all the global maximizers of a given nonlinear optimization problem. We propose a new algorithm that combines the simulated annealing (SA) method with a function stretching technique, to generate a sequence of global maximization problems that are defined whenever a new maximizer is identified. To find the global maximizers, we apply the SA algorithm to the sequence of maximization problems. Results of numerical experiments with a set of well-known test problems show that the proposed method is effective. We also compare the performance of our algorithm with other multi-global optimizers.

2. Keywords: Global optimization. Simulated annealing. Function stretching technique. Multi-global optimization.

3. Introduction
The multi-global optimization problem consists of finding all the global solutions of the following maximization problem

$$\max_{t \in T} g(t) \tag{1}$$

where $g : \mathbb{R}^n \rightarrow \mathbb{R}$ is a given multi-modal objective function and T is a compact set defined by $T = \{t \in \mathbb{R}^n : a_i \leq t_i \leq b_i, i = 1, ..., n\}$.

So, our purpose is to find all points $t^* \in T$ such that

$$\forall t \in T,\ g(t^*) \geq g(t).$$

This type of problem appears in many practical situations, for example, in ride comfort optimization [7] and in some areas of the chemical engineering (such as process synthesis, design and control) [8]. Reduction methods for solving semi-infinite programming problems also require multi-global optimizers [18, 23].

The multi-global optimization problem is a particular problem of global optimization. Thus a multi-global optimization method is an extension of a global optimizer.

The most used methods for solving a multi-global optimization problem rely on, for example, evolutionary algorithms (such as the genetic algorithm [2] and the particle swarm optimization algorithm [17]), and variants of the multi-start algorithm (clustering, domain elimination, zooming, repulsion) [26]. Another contribution can be found in [25].

The simulated annealing (SA), proposed in 1983 by Kirkpatrick, Gelatt and Vecchi, and in 1985 by Cërny, appeared as a method to solve combinatorial optimization problems. Since then, the SA algorithm has been applied in many areas such as the graph partitioning, graph coloring, number partitioning, circuit design, composite structural design, data analysis, image reconstruction, neural networks, biology, geophysics and finance [11, 13, 20]. Usually the SA method converges to just one global solution in each run.

Recently, a new technique based on function stretching has been used in a particle swarm optimization context [17], in order to avoid the premature convergence of the method to local (non-global) solutions.

In this paper, we propose to use the function stretching technique with a simulated annealing algorithm to be able to compute all the global solutions of problem (1). Each time a global maximizer is detected by the SA algorithm, the objective function of the problem is locally transformed by a function stretching that eliminates the detected maximizer leaving the other maximizers unchanged. This process is repeated until no more global solution is encountered.

*Work partially supported by FCT grant POCTI/MAT/58957/2004.

This paper is organized as follows. Section 2 describes the simulated annealing method. Section 3 contains the basic ideas behind the function stretching technique. Our proposed algorithm is presented in Section 4 and the numerical results and some conclusions are shown in Sections 5 and 6, respectively.

2. Simulated annealing method

The simulated annealing method is a well-known stochastic method for global optimization. It is also one of the most used algorithms in global optimization, mainly due to the fact that it does not require any derivative information and specific conditions on the objective function. Furthermore, it has been proved that the SA algorithm asymptotically converges to a global solution.

The SA method can be easily described using four phases: the generation of a new candidate point, the acceptance criterion, the reduction of the control parameters and the stopping criterion.

The generation of a new candidate point is one of its crucial phases and it should provide a good exploration of the search region as well as a feasible point. A generating probability density function, $f_{t^k y}(.)$, is used to find a new point y based on the current approximation, t^k. We refer to Bohachevsky *et al.* [1], Corana *et al.* [3], Szu and Hartley [22], Dekkers and Aarts [4], Romeijn and Smith [19], Ingber [11] and Tsallis and Stariolo [24] for details.

The acceptance criterion allows the SA algorithm to avoid getting stuck in local solutions when searching for a global one. For that matter, the process accepts points whenever an increase of the objective function is verified.

The acceptance criterion has the following mathematical form

$$t^{k+1} = \begin{cases} y & \text{if } \tau \leq A_{t^k y}(c_A^k) \\ t^k & \text{otherwise} \end{cases}$$

where t^k is the current approximation to the global maximum, y is the new candidate point, τ is a random number drawn from $U(0,1)$ and $A_{t^k y}(c_A^k)$ is the acceptance function. This function represents the probability of accepting the point y when t^k is the current point, and it depends on a positive control parameter c_A^k and on the difference of the function values at the points y and t^k.

The acceptance criterion based on the following acceptance function

$$A_{t^k y}(c_A^k) = \min \left\{ 1, e^{-\frac{g(t^k)-g(y)}{c_A^k}} \right\}$$

is known as Metropolis criterion. This criterion accepts all points where the objective function value increases, i.e., $g(t^k) \leq g(y)$. However, if $g(t^k) > g(y)$, the point y might be accepted with some probability. During the iterative process, the probability of descent movements decreases slowly to zero. Different acceptance criteria are proposed in Ingber [11] and Tsallis and Stariolo [24], for example.

The control parameter c_A^k, also known as temperature or cooling schedule, must be updated in order to define a positive decreasing sequence, verifying

$$\lim_{k \to \infty} c_A^k = 0.$$

When c_A^k is high, the maximization process searches in the whole feasible region, looking up for promising regions to find the global maximum. As the algorithm develops, c_A^k is slowly reduced and the algorithm computes better precision approximations to the optimum. For a good performance of the algorithm, the initial control parameter must be sufficiently high (to search for promising regions) but not extremely high because the algorithm becomes too slow. To solve this dilemma, some authors suggested that a preliminary analysis of the objective function should be done in order to get an appropriate value. For more details see Dekkers and Aarts [4], Ingber [11] and Laarhoven and Aarts [14].

All stopping criteria for the SA are based on the idea that the algorithm should terminate when no further changes occur. The usual stopping criterion limits the number of function evaluations, or defines a lower limit for the value of the control parameter. See Corana *et al.* [3], Dekkers and Aarts [4] and Ingber [11] for different alternatives.

2.1. Variant of the SA method: ASA algorithm

Adaptive Simulated Annealing (ASA) proposed by Ingber, in 1989, is today the most used variant of the SA method and it is characterized by two functions: the generating probability density function, $f_{t^k y}(c_G^k)$, and the acceptance function, $A_{t^k y}(c_A^k)$. Both functions depend on the current approximation, on the new candidate point and on the control parameters, $c_G^k \in \mathbb{R}^n$ and $c_A^k \in \mathbb{R}$, respectively.

Without getting into much details, the new candidate point, $y^T = (y_1, ..., y_n)$, is determined as follows:

$$y_i = t_i^k + \lambda_i (b_i - a_i) \text{ for } 1 \leq i \leq n \tag{2}$$

where a_i and b_i are the lower and upper bounds for the t_i variable, respectively. The value $\lambda_i \in (-1, 1)$ is given by

$$\lambda_i = \text{sign}\left(u - \frac{1}{2}\right)\left(\left(1 + \frac{1}{c_{G_i}^k}\right)^{|2u-1|} - 1\right) c_{G_i}^k \tag{3}$$

where u is a uniformly distributed random variable in $(0, 1)$ and $\text{sign}(x)$ represents the three-valued sign function

$$\text{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0. \end{cases}$$

When y is not a feasible point, the equations (2) and (3) can be repeatedly used until a feasible point is encountered. Alternatively, the point y might be projected onto the feasible region [20].

In order to update the control parameters $c_{G_i}^k$, the ASA algorithm proceeds as follows:

$$\begin{cases} k_{G_i} = k_{G_i} + 1 \\ c_{G_i}^k = c_{G_i}^0 e^{-\kappa (k_{G_i})^{\frac{1}{n}}} \end{cases} \quad \text{for } 1 \leq i \leq n \tag{4}$$

where $c_{G_i}^0$ is the initial value of the control parameter c_{G_i} and κ is defined by $\kappa = -\ln(\epsilon) e^{-\frac{\ln(N_\epsilon)}{n}}$. The values ϵ and N_ϵ should be chosen in a way that

$$\begin{cases} c_{G_i}^f = c_{G_i}^0 \epsilon \\ k^f = N_\epsilon, \end{cases}$$

being $c_{G_i}^f$ an estimate of the final value of the control parameter c_{G_i}, and k^f represents a threshold value for the maximum number of iterations. To see how the values of ϵ and N_ϵ influence the algorithm we refer the work of Niu [16].

Similarly, the control parameter c_A^k is updated by

$$\begin{cases} k_A = k_A + 1 \\ c_A^k = c_A^0 e^{-\kappa (k_A)^{\frac{1}{n}}} \end{cases} \tag{5}$$

where c_A^0 is its initial value.

To speed up the search process, this variant of the SA algorithm considers the reannealing of the process, meaning that the control parameters are redefined during the iterative process. For that, at the end of every cycle of $N_{A\_\max}$ accepted points, the algorithm evaluates certain quantities, denoted by sensitivities, which are given by

$$s_i = \left| \frac{g(t^* + \delta t_i^* e_i) - g^*}{\delta t_i^*} \right|$$

where t^* is the best point found so far, g^* represents its corresponding function value, δ is a small real parameter and $e_i \in \mathbb{R}^n$ is the i^{th} euclidian vector.

Depending on the following values

$$\rho_i = \frac{s_{\max}}{s_i} \frac{c_{G_i}^k}{c_{G_i}^0}$$

for all i, where

$$s_{\max} = \max_{1 \le i \le n} \{s_i\},$$

the quantities k_{G_i}, reported in the updating scheme (4), are redefined as follows:

$$k_{G_i} = \begin{cases} \left[-\frac{1}{\kappa} \ln(\rho_i)\right]^n & \text{if } \rho_i < 1 \\ 1 & \text{otherwise.} \end{cases}$$

As concerns the redefinition of the parameters c_A^0 and k_A in the updating scheme (5), the procedure evaluates

$$c_A^0 = \min\left\{c_A^0, \max\left\{|g(t^k)|, |g^*|, |g(t^k) - g^*|\right\}\right\}$$

and

$$k_A = \left[-\frac{1}{\kappa} \ln\left(\frac{\bar{c}_A}{c_A^0}\right)\right]^n$$

where $\bar{c}_A = \min\left\{c_A^0, \max\left\{|g(t^k) - g^*|, c_A^k\right\}\right\}$, whenever $N_{A\_\max}$ accepted points are reported.

The iterative process terminates if the found approximation to the global solution does not change for a fixed number of iterations, $M_{f^*}^{max}$, or a maximum number of function evaluations is reached, herein represented by $M_{fe}^{max} = n\,\bar{M}_{fe}^{max}$, for a threshold value $\bar{M}_{fe}^{max}$. This condition is motivated by the fact that the efficiency of ASA algorithm substantially depends on the problem dimension.

A description of the ASA algorithm follows.

ASA Algorithm

Given an initial feasible approximation t^0 and the number of iterations for reannealing $N_{A\_\max}$, compute $\kappa = -\ln(\epsilon)e^{-\frac{\ln(N_\epsilon)}{n}}$, let $k_A = k_{G_i} = 0$, calculate c_A^0, $c_{G_i}^0 = 1.0$, $n_A = 0$ and $k = 0$

while *stopping criterion is not reached* **do**

Generate a new feasible candidate point y

Analyze the acceptance criterion

if *y is accepted* **then** *set* $n_A = n_A + 1$

Set $k = k + 1$

if $n_A \ge N_{A\_\max}$ **then** *redefine* k_{G_i}, k_A *and* c_A^0, *set* $n_A = 0$

Update the control parameters $c_{G_i}^k$, k_{G_i}, c_A^k, k_A

end while

end algorithm

For details on the algorithm convergence analysis, see [11, 12].

3. Function stretching technique

For multi-modal functions, some global optimization algorithms converge prematurely to local solutions. This is the case with the simplest versions of the particle swarm optimization algorithm. To overcome this problem, Parsopoulos and Vrahatis [17] proposed a function stretching technique that provides a way to escape from local optima when the particle swarm optimization convergence stagnates, driving the search to a global solution. This technique works in the following way. When a local maximizer $\bar{t}$ is detected, a two-stage transformation of the original objective function is carried out as follows:

$$\bar{g}(t) = g(t) - \frac{\delta_1}{2}\|t - \bar{t}\|\left(\operatorname{sign}\big(g(\bar{t}) - g(t)\big) + 1\right), \tag{6}$$

$$\widetilde{g}(t) = \bar{g}(t) - \frac{\delta_2}{2} \frac{\operatorname{sign}\left(g(\bar{t}) - g(t)\right) + 1}{\tanh\left(\mu\left(\bar{g}(\bar{t}) - \bar{g}(t)\right)\right)}, \tag{7}$$

where δ_1, δ_2 and μ are positive constants.

At points t that verify $g(t) < g(\bar{t})$, the transformation defined in (6) reduces the original objective function values by $\delta_1 \|t - \bar{t}\|$. The second transformation (7) emphasizes the decrease of the original objective function by making a substantial reduction on the objective function values.

For all points t such that $g(t) \geq g(\bar{t})$, the objective function values remain unchanged, so allowing the location of the global maximizer. When applying the global algorithm to the function $\widetilde{g}$, the method is capable of finding other local solutions, $\widetilde{t}$, that satisfy $g(\widetilde{t}) \geq g(\bar{t})$. If another local (non-global) solution is found, the process is repeated until the global maximum is encountered.

Parsopoulos and Vrahatis also proposed in [17] a different version of the particle swarm optimization algorithm for locating multiple global solutions. Based on the function stretching technique (or a deflation technique), the algorithm isolates sequentially points that have objective function values larger than a threshold value, and performs a local search (with a small swarm) in order to converge to a global solution, while the big swarm continues searching the rest of the region for other global solutions.

4. Stretched simulated annealing algorithm

The Stretched Simulated Annealing (SSA) algorithm herein proposed is capable of finding all global solutions of problem (1) combining the ASA algorithm, described in Section 2, with local applications of the function stretching technique. In our case, this technique is applied not to avoid local solutions but to find all global maxima, since ASA algorithm convergence to a global solution is guaranteed with probability one. Assume now that the following assumption is verified.

Assumption 1: *All global solutions of problem (1) are isolated points.*

At each iteration, the SSA algorithm solves, using the ASA algorithm, the following global optimization problem:

$$\min_{t \in T} \Phi_k(t) \equiv \begin{cases} g(t) & \text{if } k = 1 \\ w(t) & \text{if } k > 1, \end{cases}$$

where the function $w(t)$ is defined as

$$w(t) = \begin{cases} \widetilde{g}(t) & \text{if } t \in V_\varepsilon(\bar{t}^i) \\ g(t) & \text{otherwise }, \end{cases}$$

and $\bar{t}^i$ $(i = 1, 2, ..., \bar{m})$ denotes a previously found global maximizer. $V_\varepsilon(\bar{t}^i)$ represents a neighborhood of $\bar{t}^i$, with ray ε, $\bar{m}$ is the number of desired global solutions of (1) and $\widetilde{g}$ is the function defined in (7).

The SSA algorithm resorts in a sequence of global optimization problems whose objective functions are the original g, in the first iteration, and the transformed w in the subsequent iterations. As the function stretching technique is only applied in a neighborhood of an already detected global maximizer, the ASA global algorithm is able to identify the other global maximizers that were not yet found.

To illustrate this idea, we consider the test function (herein named Parsopoulos) reported in [17],

$$\mathrm{CS}(t) = -\left(\cos^2(t_1) + \sin^2(t_2)\right)$$

with feasible region $[-5, 5]^2$. In this hypercube, the function CS has 12 global maximizers. The plot of $\mathrm{CS}(t)$ is given in the left figure of Figure 1.

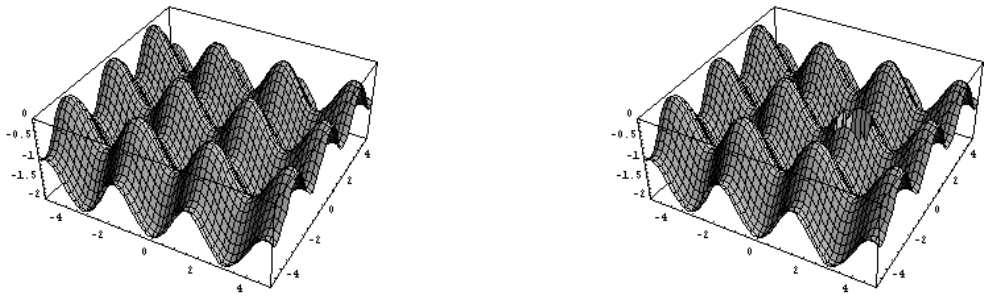

Figure 1: Original function CS and stretched CS

Applying the function stretching in a neighborhood of one global maximizer, for example $t^* = (\frac{\pi}{2}, 0)^T$, we obtain the "stretched function CS" that can be seen on the right of Figure 1. Figure 2 illustrates with more detail the application of function stretching, in the neighborhood of the optimum. The plot on the left shows the function $\bar{g}$ (obtained after the first transformation) and the one on the right represents the function $\tilde{g}$ (after the second transformation).

Figure 2: Stretched function CS: first and second transformations

As we can see, when the function stretching is applied in a neighborhood of a global maximizer, this maximum disappears and the other global solutions are left unchanged (see Figure 1). Thus, our SSA algorithm is able to detect all global solutions of problem (1).

This iterative process terminates if no new global maximizer is detected, in a fixed number of successive iterations, $N_{f^*}^{max}$, or a maximum number of function evaluations, $N_{fe}^{max} = n\bar{N}_{fe}^{max}$ is reached, for a threshold $\bar{N}_{fe}^{max}$.

5. Numerical results

The proposed algorithm was implemented in the C programming language and connected with AMPL [9] to provide the coded problems. AMPL is a mathematical programming language that allows the codification of optimization problems in a powerful and easy to learn language. AMPL also provides an interface that can be used to communicate with a solver.

For the stopping criterion of the ASA algorithm we chose the following parameters: $M_{f^*}^{max} = 5$ and $\bar{M}_{fe}^{max} = 10000$. The constants in equations (6) and (7) were set to $\delta_1 = 100$, $\delta_2 = 1$ and $\mu = 10^{-3}$. Finally, in the SSA algorithm we considered the following parameters: $N_{f^*}^{max} = 3$, $\bar{N}_{fe}^{max} = 50000$ and $\varepsilon = 0.25$. To obtain the initial control parameter c_A^0, a preliminary analysis for each test function was carried out, with a sample of $10n$ feasible points [4].

The efficiency of our SSA algorithm was tested using a set of 22 multi-modal test problems. Some of the chosen problems have more than one global maximum and the others have just one global maximum but more than ten local maxima. Table 1 reports on the main characteristics of the test problems, namely the name of the problem, the reference from where we took the problem (Ref.), the number of variables (n), the feasible region (T), the number of known global maximizers ($N_{t_g^*}$), the number of known local (non-global) maximizers ($N_{t_l^*}$) and the known global maximum value (f^*).

Table 1: Test problems.

Name	Ref.	n	T	$N_{t_g^*}$	$N_{t_l^*}$	f^*
Bohachevsky	[10]	2	$[-100, 100]^2$	1	More than 10	0
Branin	[6]	2	$[-5, 10] \times [0, 15]$	3	No local	0.397887
Cos, C_1	[15]	1	$[0, 10]$	3	No local	0
Cos, C_2	[15]	2	$[0, 10]^2$	9	No local	0
Griewank, G_2	[25]	2	$[-100, 100]^2$	1	About 530	0
Hump, H_3	[10]	2	$[-5, 5]^2$	2	No local	0
Hump, H_6	[25]	2	$[-5, 5]^2$	2	4	-1.031628
Levy, L_3	[17]	2	$[-5, 5]^2$	4	About 300	-176.541793
Levy, L_5	[17]	2	$[-10, 10]^2$	1	About 760	-176.137578
Parsopoulos	[17]	2	$[-5, 5]^2$	12	No local	0
Rastrigin	[25]	2	$[-1, 1]^2$	1	About 50	-2.0
Sines	[5]	2	$[-10, 10]^2$	1	About 50	0.9
Shubert	[2]	2	$[-10, 10]^2$	18	About 760	-186.730908
Storn, S_1	[21]	2	$[-16, 16]^2$	2	1	-0.407461
Storn, S_2	[21]	2	$[-16, 16]^2$	2	1	-18.058697
Storn, S_3	[21]	2	$[-16, 16]^2$	2	1	-227.765750
Storn, S_4	[21]	2	$[-16, 16]^2$	2	1	-2429.41477
Storn, S_5	[21]	2	$[-16, 16]^2$	2	1	-24776.5183
Storn, S_6	[21]	2	$[-16, 16]^2$	2	1	-249293.018
P_8	[4]	3	$[-10, 10]^3$	1	About 5^3	0
P_{16}	[4]	5	$[-5, 5]^5$	1	About 15^5	0
f_1	(Proposed)	2	$[-10, 10]^2$	2	No local	-100.0

Each problem was run 5 times with randomly generated initial approximations. The numerical results are shown in two separate tables. Table 2 contains the problems with more than one global maximum and Table 3 contains the problems with one global maximum and more than ten local maxima. These tables report the averaged numbers of: percentage of frequency of occurrence (Freq. Occur.), number of ASA calls (N_{ASA}), number of function evaluations (N_{FE}) and best function value (f_m^*). The last column reports the best function value obtained in all 5 runs (f^*). The percentage of frequency of occurrence is the ratio between the number of detected global maximizers and the number of known maximizers. So, 100% means that all known global maximizers were detected in all 5 runs.

Table 2: Numerical results obtained with test problems with more than one global solution.

Name	Freq. Occur.	N_{ASA}	N_{FE}	f_m^*	f^*
Branin	100%	6	10529	0.397887	0.397887
Cos, C_1	100%	6	3206	2×10^{-9}	4×10^{-13}
Cos, C_2	100%	13	19245	9×10^{-10}	3×10^{-10}
Hump, H_3	100%	5	20200	1×10^{-7}	5×10^{-8}
Hump, H_6	100%	5	17531	-1.031628	-1.031628
Levy, L_3	65%	6	13438	-176.541793	-176.541793
Parsopoulos	100%	15	16542	3×10^{-9}	3×10^{-10}
Shubert	99%	32	51684	-186.730908	-186.730908
Storn, S_1	100%	5	5850	-0.407461	-0.407461
Storn, S_2	100%	5	39877	-18.058697	-18.058697
Storn, S_3	100%	5	63510	-227.765749	-227.765750
Storn, S_4	100%	5	59841	-2429.41476	-2429.41477
Storn, S_5	100%	5	101864	-24776.5183	-24776.5183
Storn, S_6	100%	5	103191	-249293.018	-249293.018
f_1	100%	5	32572	-100.0	-100.0

The numerical results of Table 2 indicate that the SSA algorithm finds good precision approximations to the global solution, with absolute errors (difference between the average best function value and the best function value obtained in all runs) smaller than 10^{-7} (except in problems S_4 and S_6).

The runs with the test problems S_5 and S_6 stopped because the maximum number of function evaluations was reached. Nevertheless, all known global maximizers were detected in all runs. However, relaxing that limit, the SSA algorithm stops with higher precision approximations in 194325 and 180429 function evaluations respectively.

In some well-known multi-modal problems, the existence of many local maximizers makes it quite difficult for most global algorithms to determine the global solution. Usually, the algorithms stop prematurely in a local solution. We decided to analyze the behavior of our SSA algorithm in this class of problems. So, Table 3 reports the numerical results obtained with problems that have only one global solution but many local solutions.

Table 3: Numerical results obtained with problems that have only one global solution.

Name	Freq. Occur.	N_{ASA}	N_{FE}	f_m^*	f^*
Bohachevsky	100%	5	24066	2×10^{-6}	4×10^{-11}
Griewank, G_2	100%	9	39834	9×10^{-8}	11×10^{-11}
Levy, L_5	100%	4	5557	-176.137577	-176.137578
Rastrigin	100%	4	16144	-2.0	-2.0
Sines	100%	40	101445	0.900001	0.9
P_8	100%	4	4613	1×10^{-6}	6×10^{-8}
P_{16}	100%	4	15115	1×10^{-6}	8×10^{-8}

We may observe that for all tested problems the absolute error of the approximations is smaller than 10^{-6}, thus indicating that the SSA algorithm when solving these problems has a similar behavior to the one reached with the previous class of problems.

With the problems Bohachevsky, G_2, L_5 and Sines, the SSA algorithm was also able to detect some local solutions. In particular, with the function Sines the iterative process finds 28 local solutions. This happens because the function has a large number of local maxima whose values are quite similar to the value of the global maximum.

The function stretching efficiency is more evident in the problems G_2 and L_5. We run separately the ASA algorithm and obtained percentages of frequency of occurrence of 20% and 60%, respectively. When we use the SSA algorithm these percentages climb to 100% (see Table 3).

It does not seem an easy task to compare the performance of our algorithm with other multi-global solvers as some authors failed to report on important data, namely the number of function evaluations required to reach the solutions. For example, for the problem Parsopoulos that was also solved by the Stretched PSO algorithm in [17], the authors claim to find all global solutions after 12 cycles of the method with accuracy 10^{-5}, probably in a single run. No other information is reported concerning this problem. In all 5 runs, our SSA algorithm found all solutions with accuracy 10^{-9} requiring on average 15 ASA calls and 16542 function evaluations.

Meng *et al.* [15] proposed the adaptive swarm algorithm for multi-global optimization problems and presented numerical results for the three problems: C_1, C_2 and L_3. Table 4 contains a brief comparison.

Table 4: A comparison of results with those obtained by Meng *et al.*.

	Adaptive swarm algorithm			Stretched simulated annealing		
	Freq. occur.	Best solution	Worst solution	Freq. occur.	Best solution	Worst solution
C_1	93%	6.34×10^{-7}	1.05×10^{-3}	100%	4.43×10^{-13}	4.42×10^{-7}
C_2	78%	5.12×10^{-5}	3.58×10^{-2}	100%	3.09×10^{-10}	1.03×10^{-5}

For the test function L_3, Meng *et al.* only indicate that the four global solutions were obtained after

17127 function evaluations. The SSA algorithm required 14596 to reach the same solutions. For these cases, our SSA framework seems to perform favorably against the adaptive swarm algorithm.

The Differential Evolution method in [21] reaches the solution of the function Griewank with accuracy 10^{-6} with 12752 function evaluations. Our SSA algorithm requires more function evaluations but we manage to reach a better approximation. As concerns the functions $S_1, ..., S_6$, the averaged numbers of function evaluations reported in [21], for a relative accuracy of 10^{-6}, are surprisingly smaller than ours but the method finds just one global solution.

6. Conclusions

In this work, we propose a new stochastic algorithm to find all global solutions of multi-modal objective function problems. Our computational experiments show that the SSA algorithm is capable of locating all the global optima with acceptable number of function evaluations. The numerical results also indicate that the SSA algorithm is a useful tool for detecting a global optimum when the problem has a large number of local solutions.

In our view, we may adapt the SSA algorithm to find all the global solutions as well as some local (non-global) ones, probably the "best", in the sense that these local solutions have function values that satisfy

$$|g(t^*) - g(\vec{t}_i^*)| < \eta$$

where t^* represents the global maximizer and $\vec{t}_i^*$ are the desired non-global maximizers, for a fixed positive η. This issue is now under investigation.

7. References

[**1**] I. Bohachevsky, M. Johnson and M. Stein, Generalized simulated annealing for function optimization, *Technometrics*, 1986, 28(3), 209-217.

[**2**] R. Chelouah and P. Siarry, A continuous genetic algorithm designed for the global optimization of multimodal functions, *Journal of Heuristics*, 2000, 6, 191-213.

[**3**] A. Corana, M. Marchesi, C. Martini and S. Ridella, Minimizing multimodal functions of continuous variables with the simulated annealing algorithm, *ACM Transactions on Mathematical Software*, 1987, 13(3), 262-280.

[**4**] A. Dekkers and E. Aarts, Global optimization and simulated annealing, *Mathematical Programming*, 1991, 50, 367-393.

[**5**] R. Desai and R. Patil, SALO: Combining simulated annealing and local optimization for efficient global optimization, *Proc. of 9th Florida AI Research Symposium FLAIRS - 96*, 1996, 233-237.

[**6**] I. Dixon and G. Szëgo, Towards global optimisation 2, I. Dixon and G. Szëgo (Eds), *North-Holland Publishing Company*, 1978.

[**7**] P. Eriksson and J. Arora, A comparison of global optimization algorithms applied to a ride comfort optimization problem, *Structural and Multidisciplinary Optimization*, 2002, 24, 157-167.

[**8**] C. Floudas, Recent advances in global optimization for process synthesis, design and control: enclosure of all solutions, *Computers and Chemical Engineering*, 1999, 963-973.

[**9**] R. Fourer, D. Gay and B. Kernighan, A modeling language for mathematical programming, *Management Science*, 1990, 36(5), 519-554, http://www.ampl.com.

[**10**] A.-R. Hedar and M. Fukushima, Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization, *Optimization Methods and Software*, 2004, 19(3-4), 291-308.

[**11**] L. Ingber, Adaptive simulated annealing (ASA): lessons learned, *Control and Cybernetics*, 1996, 25(1), 33-54.

[**12**] L. Ingber and B. Rosen, Genetic algorithms and very fast simulated reannealing: a comparison, *Mathematical Computer Modelling*, 1992, 16(11), 87-100.

[**13**] D. Johnson, C. Aragon, L. McGeoch and C. Schevon, Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning, *Operations Research*, 1991, 39(3), 378-406.

[**14**] P. Van Laarhoven and E. Aarts, Simulated annealing: theory and applications, Mathematics and Its Applications, *Kluwer Academic Publishers*, 1987.

[**15**] T. Meng, T. Ray and P. Dhar, Supplementary material on parameter estimation using swarm algorithm, *Preprint submitted to Elsevier Science*, 2004.

[**16**] X. Niu, An integrated system of optical metrology for deep sub-micron lithography, Ph.D thesis, University of California, 1999.

[**17**] K. Parsopoulos and M. Vrahatis, Recent approaches to global optimization problems through particle swarm optimization, *Natural Computing*, 2002, 1, 235-306.

[**18**] C. Price, Non-linear semi-infinite programming, Ph.D thesis, University of Canterbury, 1992.

[**19**] H. Romeijn and R. Smith, Simulated annealing for constrained global optimization, *Journal of Global Optimization*, 1994, 5, 101-126.

[**20**] H. Romeijn, Z. Zabinsky, D. Graesser and S. Neogi, New reflection generator for simulated annealing in mixed-integer/continuous global optimization, *Journal of Optimization Theory and Applications*, 1999, 101(2), 403-427.

[**21**] R. Storn and K. Price, Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, 1997, 11, 341-359.

[**22**] H. Szu and R. Hartley, Fast simulated annealing, *Physic Letter A*, 1987, 122(3-4), 157-162.

[**23**] T. León, S. Sanmatías and H. Vercher, A multi-local optimization algorithm, *TOP*, 1998, 6(1), 1-18.

[**24**] C. Tsallis and D. Stariolo, Generalized simulated annealing, *Physics Letter A*, 1996, 233.

[**25**] I. Tsoulos and I. Lagaris, Gradient-controlled, typical-distance clustering for global optimization, *www.optimization.org*, 2004.

[**26**] W. Tu and R. Mayne, Studies of multi-start clustering for global optimization, *International Journal for Numerical Methods in Engineering*, 2002, 53, 2239-2252.